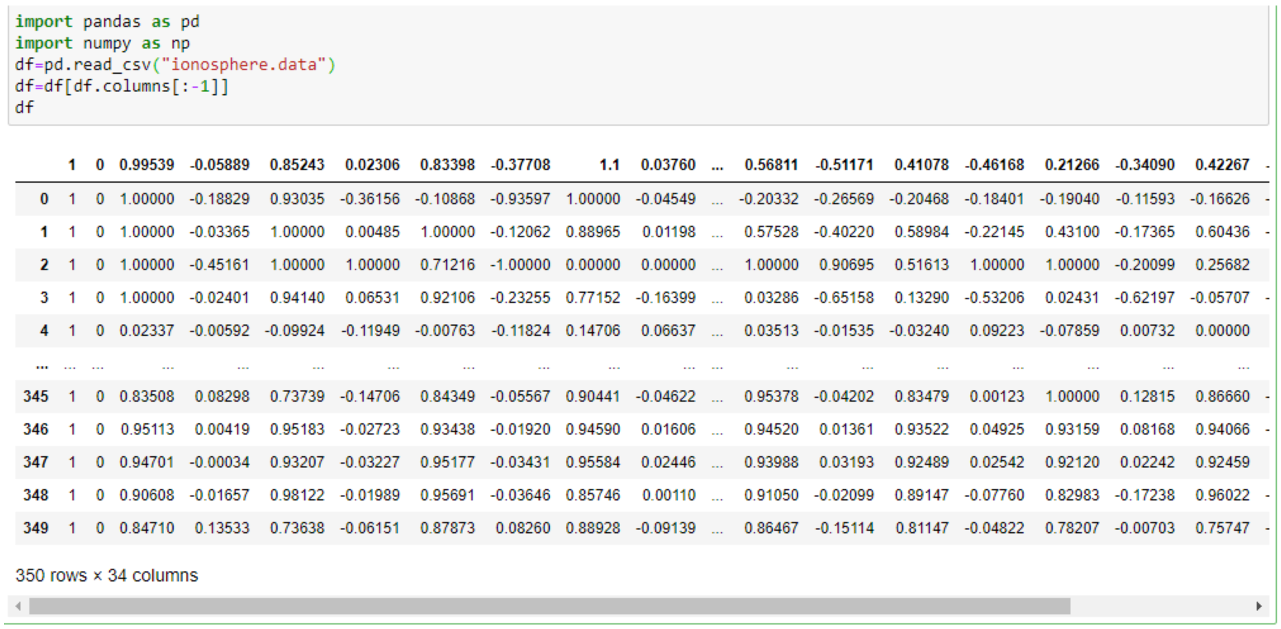
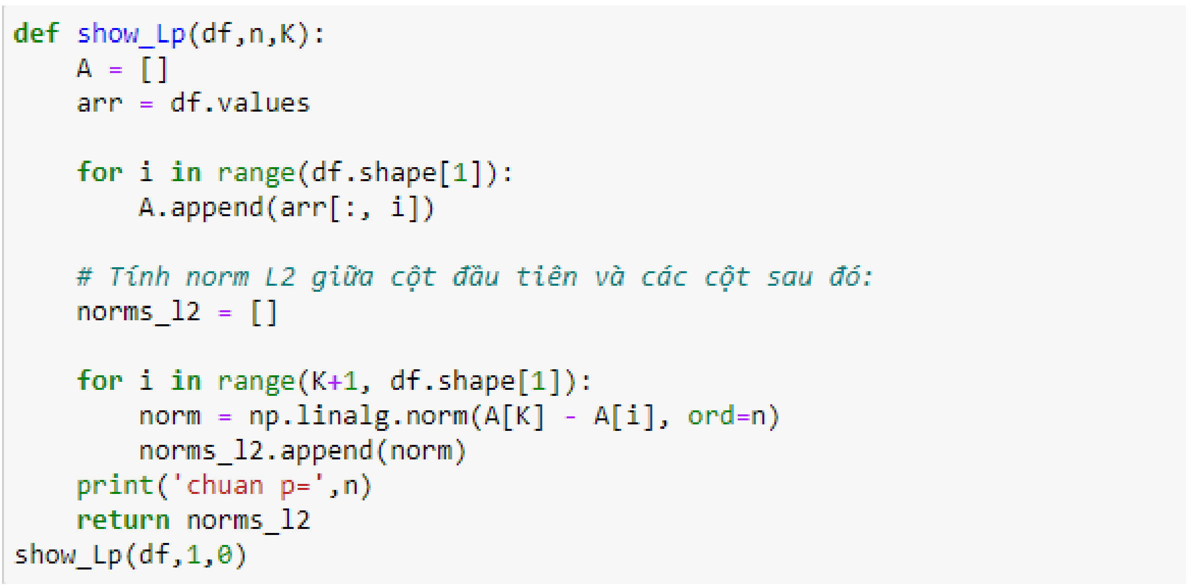


```
import pandas as pd
import numpy as np
df=pd.read_csv("ionosphere.data")
df=df[df.columns[:-1]]
df
```

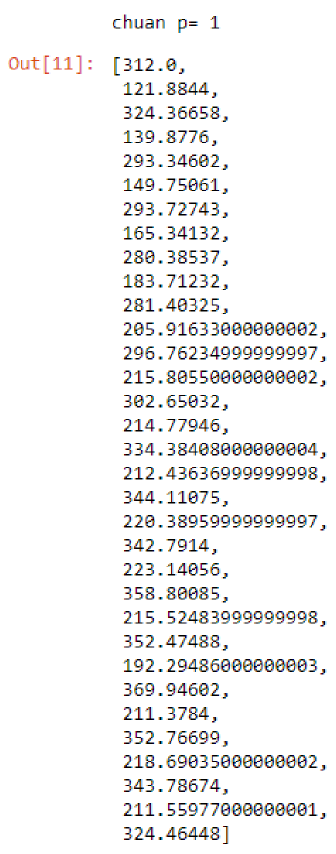
```
       1  0  0.99539  -0.05889  0.85243  0.02306  0.83398  -0.37708      1.1  0.03760  ...  0.56811  -0.51171  0.41078  -0.46168  0.21266  -0.34090  0.42267  -
  0    1  0  1.00000  -0.18829  0.93035  -0.36156  -0.10868  -0.93597  1.00000  -0.04549  ...  -0.20332  -0.26569  -0.20468  -0.18401  -0.19040  -0.11593  -0.16626  -
  1    1  0  1.00000  -0.03365  1.00000  0.00485  1.00000  -0.12062  0.88965  0.01198  ...  0.57528  -0.40220  0.58984  -0.22145  0.43100  -0.17365  0.60436  -
  2    1  0  1.00000  -0.45161  1.00000  1.00000  0.71216  -1.00000  0.00000  0.00000  ...  1.00000  0.90695  0.51613  1.00000  1.00000  -0.20099  0.25682
  3    1  0  1.00000  -0.02401  0.94140  0.06531  0.92106  -0.23255  0.77152  -0.16399  ...  0.03286  -0.65158  0.13290  -0.53206  0.02431  -0.62197  -0.05707  -
  4    1  0  0.02337  -0.00592  -0.09924  -0.11949  -0.00763  -0.11824  0.14706  0.06637  ...  0.03513  -0.01535  -0.03240  0.09223  -0.07859  0.00732  0.00000
 ...  ... ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
 345   1  0  0.83508  0.08298  0.73739  -0.14706  0.84349  -0.05567  0.90441  -0.04622  ...  0.95378  -0.04202  0.83479  0.00123  1.00000  0.12815  0.86660  -
 346   1  0  0.95113  0.00419  0.95183  -0.02723  0.93438  -0.01920  0.94590  0.01606  ...  0.94520  0.01361  0.93522  0.04925  0.93159  0.08168  0.94066  -
 347   1  0  0.94701  -0.00034  0.93207  -0.03227  0.95177  -0.03431  0.95584  0.02446  ...  0.93988  0.03193  0.92489  0.02542  0.92120  0.02242  0.92459
 348   1  0  0.90608  -0.01657  0.98122  -0.01989  0.95691  -0.03646  0.85746  0.00110  ...  0.91050  -0.02099  0.89147  -0.07760  0.82983  -0.17238  0.96022  -
 349   1  0  0.84710  0.13533  0.73638  -0.06151  0.87873  0.08260  0.88928  -0.09139  ...  0.86467  -0.15114  0.81147  -0.04822  0.78207  -0.00703  0.75747  -

350 rows × 34 columns
```

Đọc file Dataset và xóa cột và bỏ cột cuối như hướng dẫn

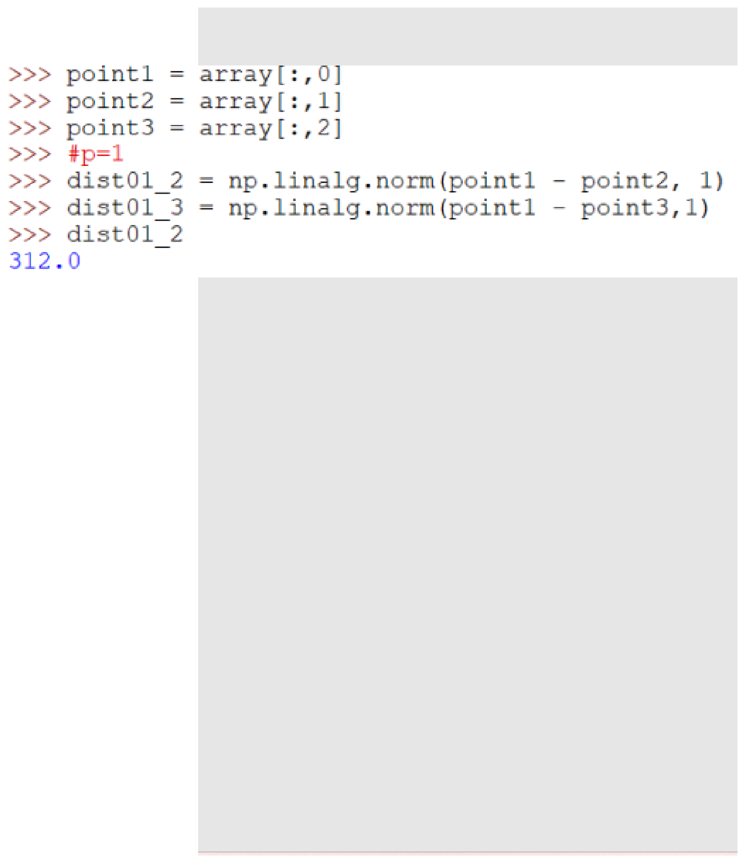


```
def show_Lp(df,n,K):
    A = []
    arr = df.values

    for i in range(df.shape[1]):
        A.append(arr[:, i])

    # Tính norm L2 giữa cột đầu tiên và các cột sau đó:
    norms_l2 = []

    for i in range(K+1, df.shape[1]):
        norm = np.linalg.norm(A[K] - A[i], ord=n)
        norms_l2.append(norm)
    print('chuan p=',n)
    return norms_l2
show_Lp(df,1,0)
```

Đưa code về 1 hàm show với df là dataset đầu vào, n tương đương chuẩn p, K đại diện cho cột thứ K trong dataframe để xét cột còn lại

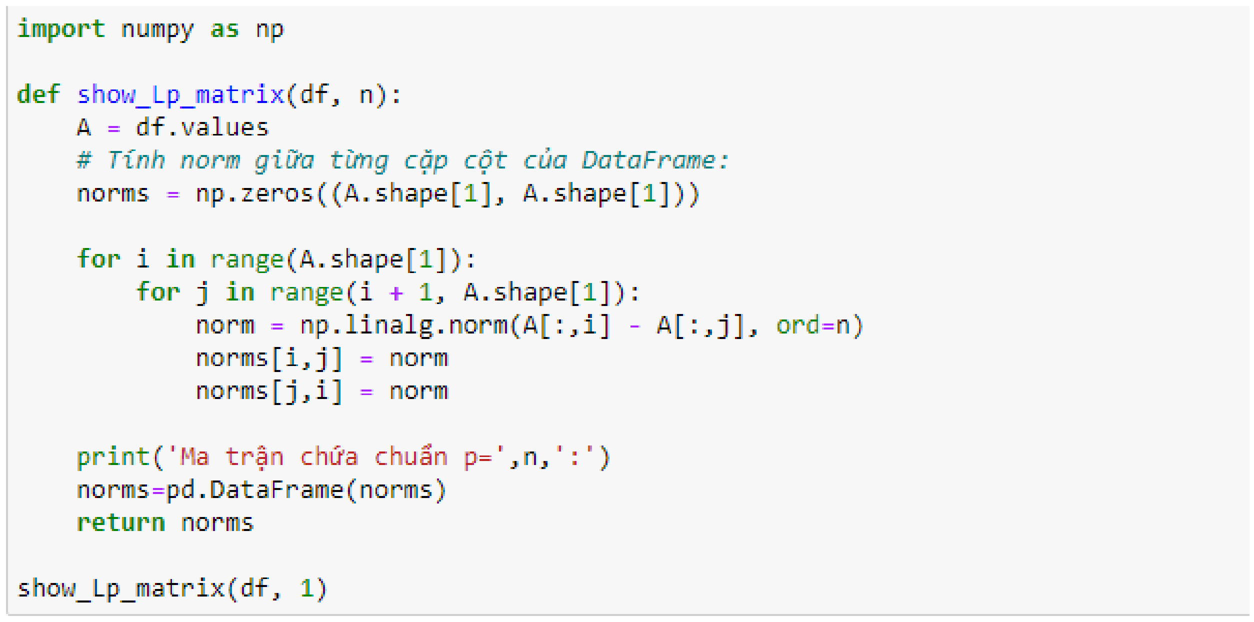


```
         chuan p= 1

Out[11]: [312.0,
          121.8844,
          324.36658,
          139.8776,
          293.34602,
          149.75061,
          293.72743,
          165.34132,
          280.38537,
          183.71232,
          281.40325,
          205.91633000000002,
          296.76234999999997,
          215.80550000000002,
          302.65032,
          214.77946,
          334.38408000000004,
          212.43636999999998,
          344.11075,
          220.38959999999997,
          342.7914,
          223.14056,
          358.80085,
          215.52483999999998,
          352.47488,
          192.29486000000003,
          369.94602,
          211.3784,
          352.76699,
          218.69035000000002,
          343.78674,
          211.55977000000001,
          324.46448]
```



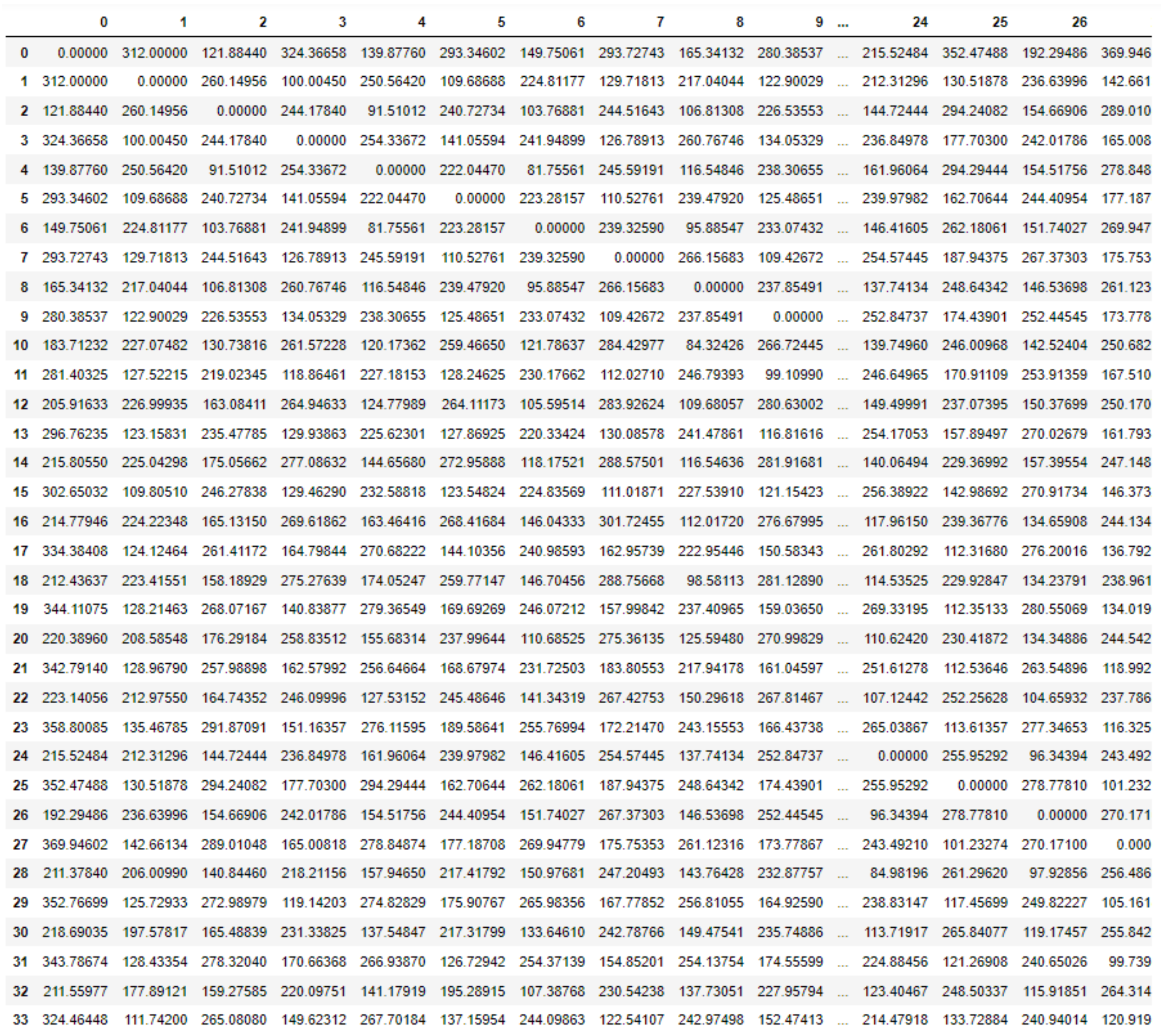
```
>>> point1 = array[:,0]
>>> point2 = array[:,1]
>>> point3 = array[:,2]
>>> #p=1
>>> dist01_2 = np.linalg.norm(point1 - point2, 1)
>>> dist01_3 = np.linalg.norm(point1 - point3,1)
>>> dist01_2
312.0
```

Nhận thấy giá trị đạt được thỏa gái trị hướng dẫn



```
import numpy as np

def show_Lp_matrix(df, n):
    A = df.values
    # Tính norm giữa từng cặp cột của DataFrame:
    norms = np.zeros((A.shape[1], A.shape[1]))

    for i in range(A.shape[1]):
        for j in range(i + 1, A.shape[1]):
            norm = np.linalg.norm(A[:,i] - A[:,j], ord=n)
            norms[i,j] = norm
            norms[j,i] = norm

    print('Ma trận chứa chuẩn p=',n,':')
    norms=pd.DataFrame(norms)
    return norms

show_Lp_matrix(df, 1)
```

Đoạn code tìm tất cả các chuẩn với p=1 của các cột với nhau

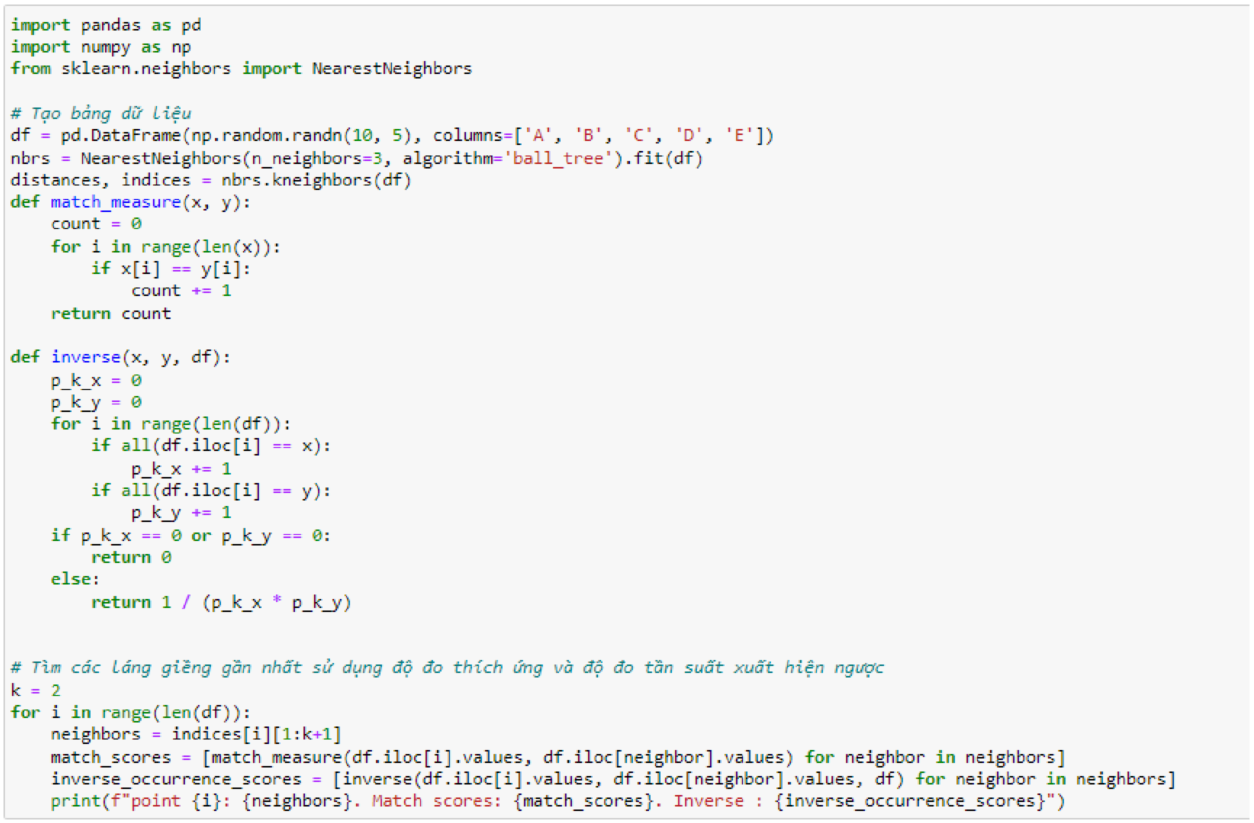


	0	1	2	3	4	5	6	7	8	9	...	24	25	26	
0	0.00000	312.00000	121.88440	324.36658	139.87760	293.34602	149.75061	293.72743	165.34132	280.38537	...	215.52484	352.47488	192.29486	369.946
1	312.00000	0.00000	260.14956	100.00450	250.56420	109.68688	224.81177	129.71813	217.04044	122.90029	...	212.31296	130.51878	236.63996	142.661
2	121.88440	260.14956	0.00000	244.17840	91.51012	240.72734	103.76881	244.51643	106.81308	226.53553	...	144.72444	294.24082	154.66906	289.010
3	324.36658	100.00450	244.17840	0.00000	254.33672	141.05594	241.94899	126.78913	260.76746	134.05329	...	236.84978	177.70300	242.01786	165.008
4	139.87760	250.56420	91.51012	254.33672	0.00000	222.04470	81.75561	245.59191	116.54846	238.30655	...	161.96064	294.29444	154.51756	278.848
5	293.34602	109.68688	240.72734	141.05594	222.04470	0.00000	223.28157	110.52761	239.47920	125.48651	...	239.97982	162.70644	244.40954	177.187
6	149.75061	224.81177	103.76881	241.94899	81.75561	223.28157	0.00000	239.32590	95.88547	233.07432	...	146.41605	262.18061	151.74027	269.947
7	293.72743	129.71813	244.51643	126.78913	245.59191	110.52761	239.32590	0.00000	266.15683	109.42672	...	254.57445	187.94375	267.37303	175.753
8	165.34132	217.04044	106.81308	260.76746	116.54846	239.47920	95.88547	266.15683	0.00000	237.85491	...	137.74134	248.64342	146.53698	261.123
9	280.38537	122.90029	226.53553	134.05329	238.30655	125.48651	233.07432	109.42672	237.85491	0.00000	...	252.84737	174.43901	252.44545	173.778
10	183.71232	227.07482	130.73816	261.57228	120.17362	259.46650	121.78637	284.42977	84.32426	266.72445	...	139.74960	246.00968	142.52404	250.682
11	281.40325	127.52215	219.02345	118.86461	227.18153	128.24625	230.17662	112.02710	246.79393	99.10990	...	246.64965	170.91109	253.91359	167.510
12	205.91633	226.99935	163.08411	264.94633	124.77989	264.11173	105.59514	283.92624	109.68057	280.63002	...	149.49991	237.07395	150.37699	250.170
13	296.76235	123.15831	235.47785	129.93863	225.62301	127.86925	220.33424	130.08578	241.47861	116.81616	...	254.17053	157.89497	270.02679	161.793
14	215.80550	225.04298	175.05662	277.08632	144.65680	272.95888	118.17521	288.57501	116.54636	281.91681	...	140.06494	229.36992	157.39554	247.148
15	302.65032	109.80510	246.27838	129.46290	232.58818	123.54824	224.83569	111.01871	227.53910	121.15423	...	256.38922	142.98692	270.91734	146.373
16	214.77946	224.22348	165.13150	269.61862	163.46416	268.41684	146.04333	301.72455	112.01720	276.67995	...	117.96150	239.36776	134.65908	244.134
17	334.38408	124.12464	261.41172	164.79844	270.68222	144.10356	240.98593	162.95739	222.95446	150.58343	...	261.80292	112.31680	276.20016	136.792
18	212.43637	223.41551	158.18929	275.27639	174.05247	259.77147	146.70456	288.75668	98.58113	281.12890	...	114.53525	229.92847	134.23791	238.961
19	344.11075	128.21463	268.07167	140.83877	279.36549	169.69269	246.07212	157.99842	237.40965	159.03650	...	269.33195	112.35133	280.55069	134.019
20	220.38960	208.58548	176.29184	258.83512	155.68314	237.99644	110.68525	275.36135	125.59480	270.99829	...	110.62420	230.41872	134.34886	244.542
21	342.79140	128.96790	257.98898	162.57992	256.64664	168.67974	231.72503	183.80553	217.94178	161.04597	...	251.61278	112.53646	263.54896	118.992
22	223.14056	212.97550	164.74352	246.09996	127.53152	245.48646	141.34319	267.42753	150.29618	267.81467	...	107.12442	252.25628	104.65932	237.786
23	358.80085	135.46785	291.87091	151.16357	276.11595	189.58641	255.76994	172.21470	243.15553	166.43738	...	265.03867	113.61357	277.34653	116.325
24	215.52484	212.31296	144.72444	236.84978	161.96064	239.97982	146.41605	254.57445	137.74134	252.84737	...	0.00000	255.95292	96.34394	243.492
25	352.47488	130.51878	294.24082	177.70300	294.29444	162.70644	262.18061	187.94375	248.64342	174.43901	...	255.95292	0.00000	278.77810	101.232
26	192.29486	236.63996	154.66906	242.01786	154.51756	244.40954	151.74027	267.37303	146.53698	252.44545	...	96.34394	278.77810	0.00000	270.171
27	369.94602	142.66134	289.01048	165.00818	278.84874	177.18708	269.94779	175.75353	261.12316	173.77867	...	243.49210	101.23274	270.17100	0.000
28	211.37840	206.00990	140.84460	218.21156	157.94650	217.41792	150.97681	247.20493	143.76428	232.87757	...	84.98196	261.29620	97.92856	256.486
29	352.76699	125.72933	272.98979	119.14203	274.82829	175.90767	265.98356	167.77852	256.81055	164.92590	...	238.83147	117.45699	249.82227	105.161
30	218.69035	197.57817	165.48839	231.33825	137.54847	217.31799	133.64610	242.78766	149.47541	235.74886	...	113.71917	265.84077	119.17457	255.842
31	343.78674	128.43354	278.32040	170.66368	266.93870	126.72942	254.37139	154.85201	254.13754	174.55599	...	224.88456	121.26908	240.65026	99.739
32	211.55977	177.89121	159.27585	220.09751	141.17919	195.28915	107.38768	230.54238	137.73051	227.95794	...	123.40467	248.50337	115.91851	264.314
33	324.46448	111.74200	265.08080	149.62312	267.70184	137.15954	244.09863	122.54107	242.97498	152.47413	...	214.47918	133.72884	240.94014	120.919

Câu 2

Vì dataset quá lớn

Chạy thử với 1 tập dataframe tạo ngẫu nhiên

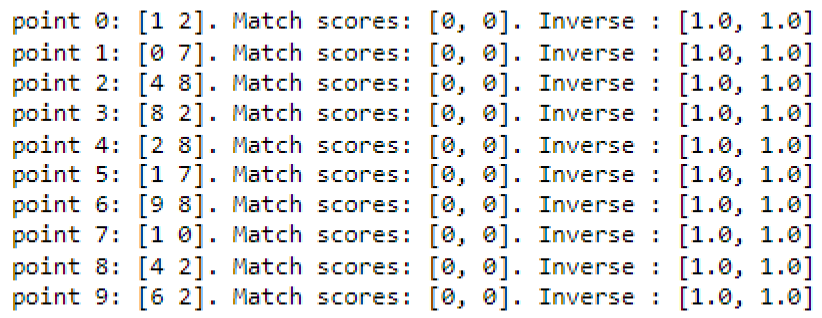


```python
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors

# Tạo bảng dữ liệu
df = pd.DataFrame(np.random.randn(10, 5), columns=['A', 'B', 'C', 'D', 'E'])
nbrs = NearestNeighbors(n_neighbors=3, algorithm='ball_tree').fit(df)
distances, indices = nbrs.kneighbors(df)
def match_measure(x, y):
    count = 0
    for i in range(len(x)):
        if x[i] == y[i]:
            count += 1
    return count

def inverse(x, y, df):
    p_k_x = 0
    p_k_y = 0
    for i in range(len(df)):
        if all(df.iloc[i] == x):
            p_k_x += 1
        if all(df.iloc[i] == y):
            p_k_y += 1
    if p_k_x == 0 or p_k_y == 0:
        return 0
    else:
        return 1 / (p_k_x * p_k_y)


# Tìm các láng giềng gần nhất sử dụng độ đo thích ứng và độ đo tần suất xuất hiện ngược
k = 2
for i in range(len(df)):
    neighbors = indices[i][1:k+1]
    match_scores = [match_measure(df.iloc[i].values, df.iloc[neighbor].values) for neighbor in neighbors]
    inverse_occurrence_scores = [inverse(df.iloc[i].values, df.iloc[neighbor].values, df) for neighbor in neighbors]
    print(f"point {i}: {neighbors}. Match scores: {match_scores}. Inverse : {inverse_occurrence_scores}")
```

Áp dụng công thức

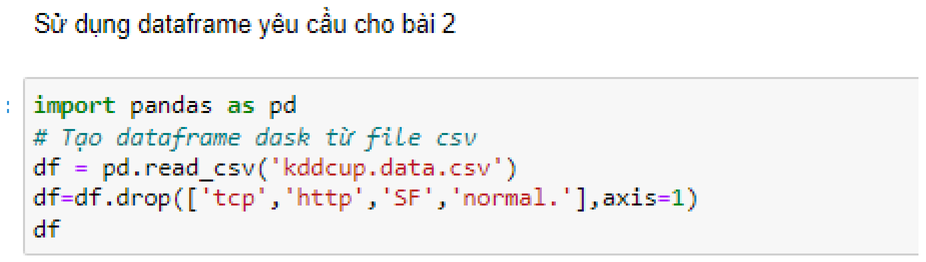
match measure: $Sim(\bar{X}, \bar{Y}) = \sum S(xi, yi)$ với $S(xi, yi)$ là sự tương đồng giữa các giá trị thuộc tính xi, yi . Lựa chọn đơn giản nhất cho $S(xi, yi)$ là $S(xi, yi) = 1$ $nếu$ $xi = yi$ và 0 $ngược$ $lại$

Độ đo tần suất xuất hiện ngược: $S(xi, yi) = 1/(pk(x)\hat{}2)$ $nếu$ $xi = yi$ và 0 $ngược$ $lại$



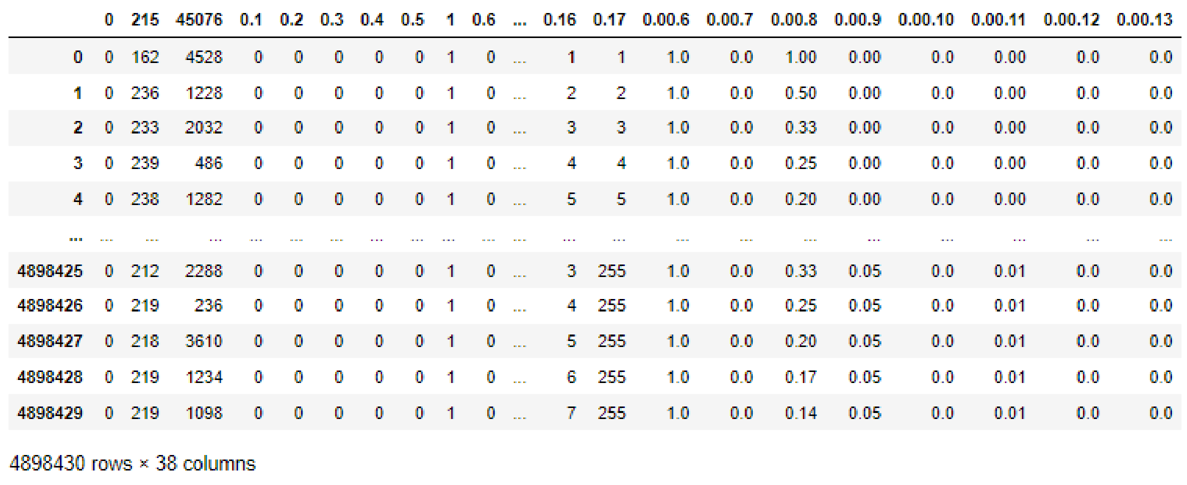
```
point 0: [1 2]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 1: [0 7]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 2: [4 8]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 3: [8 2]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 4: [2 8]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 5: [1 7]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 6: [9 8]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 7: [1 0]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 8: [4 2]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
point 9: [6 2]. Match scores: [0, 0]. Inverse : [1.0, 1.0]
```

Đọc dataset yêu cầu KDD Cup Network



Sử dụng dataframe yêu cầu cho bài 2

```
import pandas as pd
# Tạo dataframe dask từ file csv
df = pd.read_csv('kddcup.data.csv')
df=df.drop(['tcp','http','SF','normal.'],axis=1)
df
```

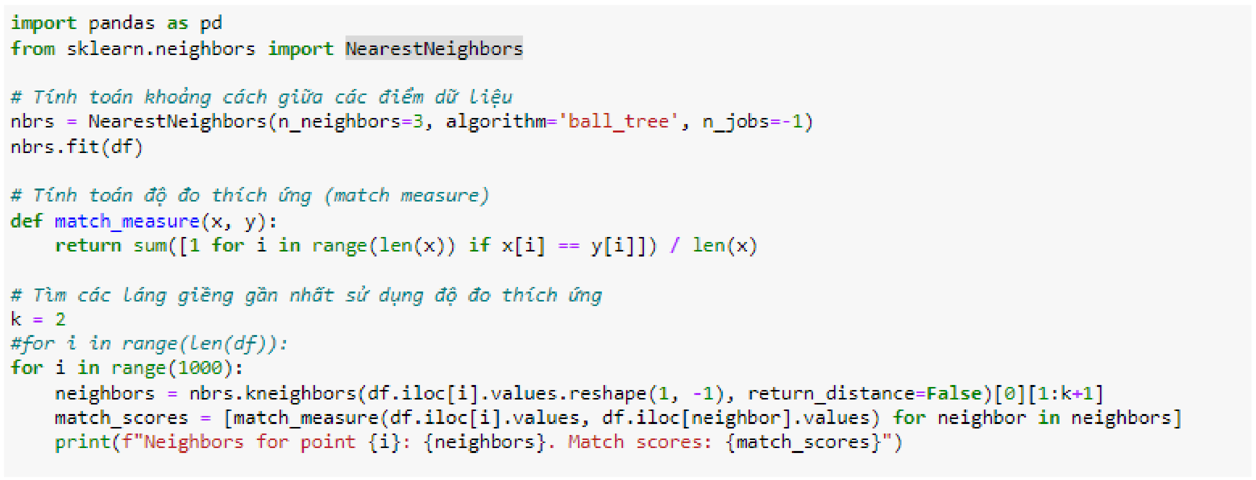


	0	215	45076	0.1	0.2	0.3	0.4	0.5	1	0.6	...	0.16	0.17	0.00.6	0.00.7	0.00.8	0.00.9	0.00.10	0.00.11	0.00.12	0.00.13
0	0	162	4528	0	0	0	0	0	1	0	...	1	1	1.0	0.0	1.00	0.00	0.0	0.00	0.0	0.0
1	0	236	1228	0	0	0	0	0	1	0	...	2	2	1.0	0.0	0.50	0.00	0.0	0.00	0.0	0.0
2	0	233	2032	0	0	0	0	0	1	0	...	3	3	1.0	0.0	0.33	0.00	0.0	0.00	0.0	0.0
3	0	239	486	0	0	0	0	0	1	0	...	4	4	1.0	0.0	0.25	0.00	0.0	0.00	0.0	0.0
4	0	238	1282	0	0	0	0	0	1	0	...	5	5	1.0	0.0	0.20	0.00	0.0	0.00	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4898425	0	212	2288	0	0	0	0	0	1	0	...	3	255	1.0	0.0	0.33	0.05	0.0	0.01	0.0	0.0
4898426	0	219	236	0	0	0	0	0	1	0	...	4	255	1.0	0.0	0.25	0.05	0.0	0.01	0.0	0.0
4898427	0	218	3610	0	0	0	0	0	1	0	...	5	255	1.0	0.0	0.20	0.05	0.0	0.01	0.0	0.0
4898428	0	219	1234	0	0	0	0	0	1	0	...	6	255	1.0	0.0	0.17	0.05	0.0	0.01	0.0	0.0
4898429	0	219	1098	0	0	0	0	0	1	0	...	7	255	1.0	0.0	0.14	0.05	0.0	0.01	0.0	0.0

4898430 rows × 38 columns

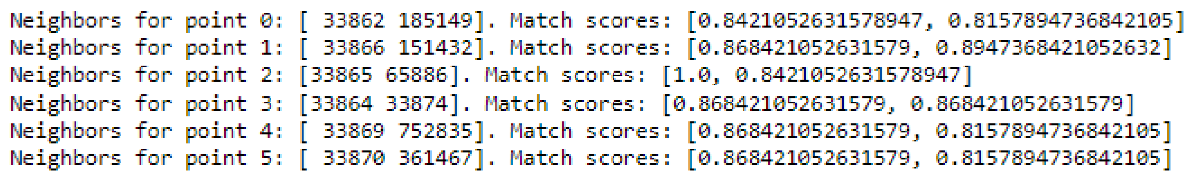
Nhận thấy số dòng quá lớn, áp dụng vòng lặp for với độ phức tạp là không khả thi, tốn nhiều thời gian

⇨ Áp dụng NearestNeighbors để tối ưu hóa thuật toán

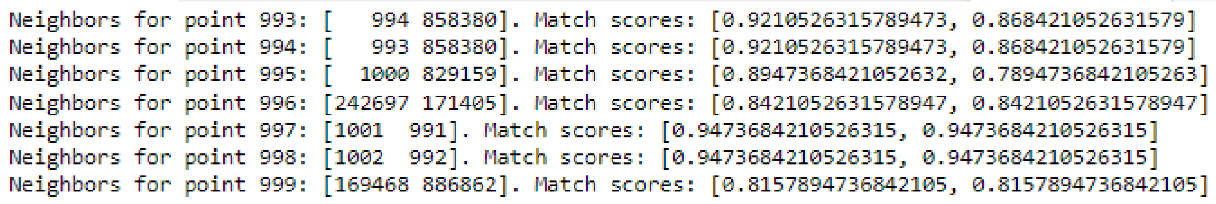


```
import pandas as pd
from sklearn.neighbors import NearestNeighbors

# Tính toán khoảng cách giữa các điểm dữ liệu
nbrs = NearestNeighbors(n_neighbors=3, algorithm='ball_tree', n_jobs=-1)
nbrs.fit(df)

# Tính toán độ đo thích ứng (match measure)
def match_measure(x, y):
    return sum([1 for i in range(len(x)) if x[i] == y[i]]) / len(x)

# Tìm các láng giềng gần nhất sử dụng độ đo thích ứng
k = 2
#for i in range(len(df)):
for i in range(1000):
    neighbors = nbrs.kneighbors(df.iloc[i].values.reshape(1, -1), return_distance=False)[0][1:k+1]
    match_scores = [match_measure(df.iloc[i].values, df.iloc[neighbor].values) for neighbor in neighbors]
    print(f"Neighbors for point {i}: {neighbors}. Match scores: {match_scores}")
```

Đầu ra gần 4900000 là quá lớn nên chỉ xuất ra 1000 kết quả được thể hiển qua vòng lặp for



```
Neighbors for point 0: [ 33862 185149]. Match scores: [0.8421052631578947, 0.8157894736842105]
Neighbors for point 1: [ 33866 151432]. Match scores: [0.868421052631579, 0.8947368421052632]
Neighbors for point 2: [33865 65886]. Match scores: [1.0, 0.8421052631578947]
Neighbors for point 3: [33864 33874]. Match scores: [0.868421052631579, 0.868421052631579]
Neighbors for point 4: [ 33869 752835]. Match scores: [0.868421052631579, 0.8157894736842105]
Neighbors for point 5: [ 33870 361467]. Match scores: [0.868421052631579, 0.8157894736842105]
```



```
Neighbors for point 993: [   994 858380]. Match scores: [0.9210526315789473, 0.868421052631579]
Neighbors for point 994: [   993 858380]. Match scores: [0.9210526315789473, 0.868421052631579]
Neighbors for point 995: [  1000 829159]. Match scores: [0.8947368421052632, 0.7894736842105263]
Neighbors for point 996: [242697 171405]. Match scores: [0.8421052631578947, 0.8421052631578947]
Neighbors for point 997: [1001  991]. Match scores: [0.9473684210526315, 0.9473684210526315]
Neighbors for point 998: [1002  992]. Match scores: [0.9473684210526315, 0.9473684210526315]
Neighbors for point 999: [169468 886862]. Match scores: [0.8157894736842105, 0.8157894736842105]
```